Next Generation Software Development

Final Report
ARO Contract DAAD 19-01-1-0723
Period covered: 08/01/01 - 01/31/05

P.I.: Prof. Zohar Manna
Computer Science Department
Stanford University
Stanford, CA. 94305-9045

April 2005

Contents

1 Statement of the Problem Studied

Under this grant we have studied the development of a scientifically sound basis for software development that builds on widely used pragmatic methods but is firmly grounded in well-established formal domains such as first-order logic and automata theory. To be sufficiently expressive for software systems, the work has focused on methods applicable to infinite-state systems. Traditionally methods for infinite-state systems have been expensive, because they were mainly deductive and thus required guidance by users who were both experts in the application domain and in the verification methodology.

Our research has been directed at algorithmic-deductive techniques that separate the combinatorial reasoning from reasoning about the data. These methods often limit user input to providing abstract system models and application-level guidance, making the interaction more natural to software developers. Constructed proofs hide low-level details; instead, they reason at the most appropriate level of abstraction with respect to the properties to be proved. This characteristic of proofs make them suitable as system documentation that can evolve with the system. To ensure well-defined semantics, computational models were developed for new computing paradigms, including aspects of publish-subscribe systems and middleware design patterns.

2 Summary of Results

2.1 Static Analysis

We have made significant contributions to the automatic construction of proofs of sequential and reactive systems by developing a new approach to invariant generation and program termination analysis.

2.1.1 Invariant Generation

Generating system invariants is one of the most important components of any verification methodology. In addition, invariants provide insight into the system. Invariant generation has been studied extensively for several decades. The traditional approaches have relied on forward propagation in an abstract interpretation framework. Starting from the initial condition, symbolic simulation is performed in an abstract domain until a fixed point is reached. The fixed point is the most precise invariant of the system representable in the abstract domain chosen. For the most popular abstract domains, however, convergence may not be reached in a finite number of steps, and one has to resort to *widening* to force convergence, thereby making the invariant less precise and potentially trivial. Despite much study, widening has remained an art more than a science, with little control of the user over the resulting precision, and often unpredictable results in practice.

The approach that we have developed addresses these problems by posing the invariant-generating problem as a constraint-solving problem [CSS03]. The conditions for an expression of a certain type to be an invariant are encoded as a constraint system and the solutions of this constraint system represent all invariants of that type. This constraint-based approach has several advantages over the traditional approach, especially for software engineering practice.

Controlling Complexity Invariant generation is inherently a hard problem with high complexity. With our method, however, the user can make deliberate choices how to trade off precision versus complexity by choosing the shape of the target invariant and strengthening the conditions on the properties to be found. In the traditional approach user control essentially ends with the choice of the abstract domain. Also, our approach allows exploitation of additional structure in the system to reduce the complexity of the constraint solving. For example in [SSM03] we showed that systems presented as Petri nets gave rise to linear constraint systems rather than nonlinear ones.

New Abstract Domains In the traditional approach invariant generation was studied mostly in the abstract domain of linear inequalities. Attempts to extend it to other domains were largely unsuccessful. With our constraint-based approach, however, target template invariants can be chosen in any domain that allows the encoding of the conditions in a (decidable) constraint system. For example, we have succeeded in generating nonlinear invariants (polynomial equalities) [SSM04] and are currently investigating application to recursive datatypes. In software engineering practice this approach creates the opportunity to develop invariant generating methods specialized for the application and its specific data structures.

New Target Properties Traditional invariant generation methods are by their very nature, forward symbolic simulation, limited to generating invariants. In the constraint-based approach, any property that can be encoded as a constraint system can be generated. For example, we have applied exactly the same techniques to the generation of ranking functions, described below. In software engineering practice, this can easily be extended to application-specific properties. We are currently developing methods to generate verification diagrams automatically.

Constraint Solvers as Engines The main bottleneck in the constraint-based approach is the complexity of solving the constraints. Constraint-solving, however, is an independent and very active area of research with many other applications. The advantage of the constraint-based approach is that any advances in constraint solving can directly be exploited by our methods. Stronger constraint solvers translate directly into improved precision and increased scalability.

2.1.2 Termination Analysis

Guaranteed termination of program loops is necessary in many settings, such as embedded systems and safety critical software. Although termination analysis methods have been studied extensively in logic and functional programming and term rewriting systems, termination of imperative programs had, until recently, received little attention.

Over the last four years we have developed a systematic, constraint-based approach towards termination analysis of imperative programs [CS01, CS02, BMS05]. In general, termination of loops is proved by exhibiting a ranking function, that is, a function that is well-founded and decreasing with each pass through the loop. We have developed methods for automatically

synthesizing such ranking functions. Like for invariant generation, the conditions for expressions of a certain type to be a ranking function are encoded as a constraint system. If the resulting constraint system is satisfiable, a ranking function exists, and hence the loop has been proved to terminate. Unlike for invariant generation, the constraint systems generated for ranking functions are linear, and hence can be solved very efficiently. Therefore this method scales remarkably well. We have demonstrated our methods on tens of thousands of lines of code, with analysis times on the order of seconds.

2.1.3 Static Analysis Tools

The lack of integration between prototype implementations of results of research blocks progress toward direct application of formal methods research in software engineering settings. We have surveyed a host of tools, examining how their integration would increase their power and benefit future research and application [BSSM04].

2.2 Dynamic Analysis

Although static analysis methods have improved considerably, in many cases they still do not scale to large software projects. A practical alternative is then to do dynamic analysis, that is, to monitor the running program. Another situation in which dynamic analysis may be the only option is when the program must run in an environment that does not tolerate violations of the specification, but the source code is not available for inspection for proprietary reasons or due to outsourcing.

Dynamic analysis tends to have lower complexity than static analysis, because runs of the system are analyzed individually, while with static analysis all (usually infinitely many) possible runs must be covered. We have developed efficient methods for runtime verification based on alternating automata [FS04, FSS02]. These methods are not limited to checking temporal properties, but can also collect runtime statistics. These statistics are useful as early warnings of impending problems in performance and resource usage.

Dynamic analysis methods can also be used as an alternative to testing. In collaboration with Synopsys we have developed a specification language and algorithms for the online and offline monitoring of synchronous systems including circuits and embedded systems [DSS+05]. The specification language can describe both correctness/failure assertions and statistical measures that are useful for system profiling and coverage analysis.

2.3 Computational Models

New computational paradigms require well-defined computational models as a basis for reasoning about systems developed according to these patterns and principles. Often, new paradigms emerge and are adopted because they are found to solve recurring problems conveniently. Modeling and analysis come afterward. We have modeled and analyzed the following new paradigms:

Aspect-oriented programming Aspect-oriented programming [KLM+97] allows component-based development with orthogonal concerns such as security or concurrency control developed and incorporated separately. We developed a computational model for aspect-oriented construction of reactive systems, which allows analysis of preservation of temporal properties [Sip03] based on early work in this area by Katz [Kat93].

Publish-subscribe Systems The publish-subscribe paradigm has emerged as a convenient architectural principle to construct large loosely-coupled distributed systems. Components publish messages to the middleware, which distributes them to components that have expressed interest by means of subscriptions. Subscriptions can be in the form of simple filters or as more complex temporal patterns, also known as event correlation expressions. In safety critical systems, but also in many other systems such as stock-trading systems, it is essential that all relevant messages are delivered to their target audiences. Practical experience with a popular, open-source middleware platform showed that it is extremely hard to get the semantics of an event-correlation service correct without a very careful analysis.

We developed an event-correlation language and defined its operational semantics in terms of concurrent automata [SSS+03]. This semantics not only allows analysis of event correlation expressions [SSSM05], but also enables automatic synthesis of the the correlators to be embedded in the middleware, thus obtaining correct-by-construction implementations.

Design Patterns Design patterns [GHJV95] provide schematic, informal solutions to frequently occurring problems in software development. Formalizing such patterns is an important first step in enabling formal analysis of various aspects of systems developed according to these

patterns. Together with middleware experts from Washington University in St Louis, we formalized one such design pattern for middleware systems, known as *WaitonConnection* [SSRB00]. We modeled remote invocations handled in accordance with this pattern and analyzed resource requirements and potential for deadlock for different thread allocation protocols [SSS+05]. This formalization can serve as an example for modeling other design patterns.

2.4 Decision Procedures

Efficient decision procedures are the cornerstone of every verification system. Decision procedures for individual theories, however, are of limited use in verification and especially in software verification, because most verification conditions involve data types that span multiple theories, for example recursive data types combined with integers.

We have developed combination methods for various data types with integers, including recursive data types and queues [ZSM04a, ZSM04b, ZSM05].

2.5 Case Studies: CARA

Automation of medical devices can save lives. It can assure continuous monitoring and control and consistent care when trained medical personnel is not available or in short supply. On the other hand, errors in the software or missed conditions can be fatal. The Food and Drug Administration (FDA) has the task to determine whether a medical device is safe for deployment. Currently this is done by extensive review of the development and testing process; no formal verification is mandated to obtain approval.

As software gets more and more complex there is a feeling that the current review process may not be adequate, and more formal techniques are called for. The CARA case study was initiated by the FDA, in collaboration with the Army Research Office, to assess the feasibility of doing formal verification for such a device

The CARA case study concerns a computer-assisted resuscitation pump used to provide fluids to people who suffer severe loss of blood, to stabilize their blood pressure until more permanent remedial actions such as surgery can be provided. The device was developed by the Walter Reid Army Institute of Research (WRAIR) in Silver Spring, MD.

We were provided with a set of tagged requirements, developed by medical experts at WRAIR. The same document was also given to the software engineers charged with developing the software. We participated in several

meetings and rounds of questions and answers with the medical expert and software engineers at WRAIR.

We modeled the system using *clocked transition systems* [MP95, KMP96, KMP98], an extension of fair transition systems to account for continuous real time. In the course of modeling this system we added several constructs to make the description more natural. Clocked transition systems are a very expressive model, and thus most questions about them are undecidable. However, the model under construction did not use the full expressiveness, and thus we were able to specialize our analysis techniques for this case, up to the point where model checking could be used for some of it. The analysis techniques were implemented in our verification tool STeP (Stanford Temporal Prover) [BBC+00].

2.6 Tool Development: STeP + AutoFocus

We have collaborated with researchers from the group of Prof. Manfred Broy of the Technical University of Munich on the integration of STeP with AutoFocus. Dr. Heiko Lötzbeyer and Alexander Wisspeintner from the Technical University of Munich visited SRI and Stanford for several weeks. We explored the integration of design and verification models between AutoFocus [HSSS96] and STeP [BBC+00]. We translated an existing AutoFocus model into STeP, while carefully considering both the structural and the behavioral aspects of the model. Extra variables had to be introduced to preserve the structural view, consisting of a network of components connected by communication channels, and its execution semantics. The behavioral view could directly be modeled by STeP transition systems, with the addition of a global clock to maintain the discrete time semantics of AutoFocus.

3 Publications

3.1 Journal papers

1. Bernd Finkbeiner and Henny B. Sipma. Checking finite traces using alternating automata. *Formal Methods in System Design*, 24:101–127, 2004. A preliminary version appeared in *Runtime Verification (RV'01), Electronic Notes in Theoretical Computer Science, Vol 55, No. 2, pp. 44–60, Elsevier Science Publishers (2001).*

3.2 Conference papers

1. Michael Colón, Sriram Sankaranarayanan, and Henny Sipma. Linear invariant generation using non-linear constraint solving. In *Computer Aided Verification*, volume 2725 of *LNCS*, pages 420–433. Springer-Verlag, July 2003.

2. Michael Colón and Henny Sipma. Practical methods for proving program termination. In *Proc.* 14^{th} *Intl. Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 442–454. Springer Verlag, 2002.

3. Bernd Finkbeiner, Sriram Sankaranarayanan, and Henny Sipma. Collecting statistics over runtime executions. In Klaus Havelund and Grigore Rosu, editors, *Proc. of Runtime Verification 2002*, volume 70 of *Electronic Notes in Theoretical Computer Science*, pages 36–55. Elsevier Science Publishers, 2002. Accepted for publication in Formal Methods in System Design.

4. Zohar Manna and Calogero Zarba. Combining decision procedures. In Bernhard K. Aichernig and Tom Maibaum, editors, *Formal Methods at the Crossroads: From Panacea to Foundational Support*, volume 2757 of *LNCS*, pages 381–422. Springer Verlag, 2003.

5. Cesar Sanchez, Sriram Sankaranarayanan, Henny B. Sipma, Ting Zhang, David L. Dill, and Zohar Manna. Event correlation: Language and semantics. In *Embedded Software (EMSOFT)*, volume 2855 of *LNCS*, pages 323–339, 2003.

6. Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Petri net analysis using invariant generation. In Nachum Derschowitz, editor, *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 682–701, Taurmina, Italy, 2003. Springer Verlag.

7. Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Constraint-based linear relations analysis. In 11^{th} *Static Analysis Symposium (SAS'2004)*, volume 3148 of *LNCS*, pages 53–68. Springer-Verlag, 2004.

8. Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Non-linear loop invariant generation using Gröbner bases,. In 31^{th} *ACM Symp. Princ. of Prog. Lang.*, pages 318–329, Venice, Italy, January 2004.

9. Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Scalable analysis of linear systems using mathematical programming. In Radhia Cousot, editor, *Proc. of Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 3385 of *LNCS*, Paris, France, January 2005. Springer Verlag.

10. Henny B. Sipma. A formal model for cross-cutting modular transition systems. In Gary Leavens and Curtis Clifton, editors, *Proceedings of the Workshop on Foundations of Aspect-Oriented Languages (FOAL'03)*, pages 9–16, 2003.

11. Ting Zhang, Henny Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints. In *the* 2^{nd} *International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *LNCS*, pages 152–167. Springer-Verlag, 2004 (Best Paper Award).

12. Ting Zhang, Henny Sipma, and Zohar Manna. Term algebras with length function and bounded quantifier alternation. In *the* 17^{th} *International Conference on Theorem Proving in Higher Order Logics (TPHOLs'04)*, volume 3223 of *LNCS*, pages 321–336. Springer-Verlag, 2004.

3.3 Manuscripts Submitted

1. Ben D'Angelo, Sriram Sankaranarayanan, César Sánchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, and Zohar Manna. Lola: Runtime monitoring of synchronous systems. In *TIME'05*, 2005. To appear.

2. Aaron R. Bradley, Henny B. Sipma, Sarah Solter, and Zohar Manna. Integrating tools for practical software analysis. In *Proc. of the Mon-*

terey Workshop. Software Engineering Tools: Compatibility and Integration, 2004. To appear.

3. Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Linear ranking with reachability. In *Proc. 17th Intl. Conference on Computer Aided Verification*, LNCS. Springer Verlag, 2005. To appear.

4. César Sánchez, Henny B. Sipma, Matteo Slanina, and Zohar Manna. Final semantics for event-pattern reactive programs. In *CALCO'05*, 2005. To appear.

5. César Sanchez, Henny B. Sipma, Venkita Subramonian, Christopher Gill, and Zohar Manna. Thread allocation protocols for distributed and real-time and embedded systems. 2005. Submitted.

6. Ting Zhang, Henny Sipma, and Zohar Manna. Decision procedures for queues with integer constraints, 2005. Submitted.

4 Scientific Personnel

The following people have been involved with the project:

Prof. Zohar Manna (P.I.)
Dr. Henny B. Sipma (Sr. Research Associate)
Dr. Bernd Finkbeiner (PhD, 2002)
Dr. Michael Colon (PhD, 2003)
Dr. Calogero Zarba (PhD, 2004)
Ben D'Angelo (undergraduate student)
Aaron Bradley (PhD student)
César Sánchez (PhD student)
Sriram Sankaranarayanan (PhD student)

References

[BBC+00] Nikolaj S. Bjørner, Anca Browne, Michael Colón, Bernd Finkbeiner, Zohar Manna, Henny B. Sipma, and Tomás E. Uribe. Verifying temporal properties of reactive systems: A STeP tutorial. *Formal Methods in System Design*, 16(3):227–270, June 2000.

[BMS05] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Linear ranking with reachability. In *Proc. 17th Intl. Conference on Computer Aided Verification*, LNCS. Springer Verlag, 2005. To appear.

[BSSM04] Aaron R. Bradley, Henny B. Sipma, Sarah Solter, and Zohar Manna. Integrating tools for practical software analysis. In *Proc. of the Monterey Workshop. Software Engineering Tools: Compatibility and Integration*, 2004. To appear.

[CS01] Michael Colón and Henny Sipma. Synthesis of linear ranking functions. In Tiziana Margaria and Wang Yi, editors, *7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2031 of *LNCS*, pages 67–81. Springer Verlag, April 2001.

[CS02] Michael Colón and Henny Sipma. Practical methods for proving program termination. In *Proc. 14th Intl. Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 442–454. Springer Verlag, 2002.

[CSS03] Michael Colón, Sriram Sankaranarayanan, and Henny Sipma. Linear invariant generation using non-linear constraint solving. In *Computer Aided Verification*, volume 2725 of *LNCS*, pages 420–433. Springer-Verlag, July 2003.

[DSS+05] Ben D'Angelo, Sriram Sankaranarayanan, César Sánchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, and Zohar Manna. Lola: Runtime monitoring of synchronous systems. In *TIME'05*, 2005. To appear.

[FS04] Bernd Finkbeiner and Henny B. Sipma. Checking finite traces using alternate automata. *Formal Methods in System Design*, 24:101–127, 2004. A preliminary version appeared in *Runtime*

Verification (RV'01), Electronic Notes in Theoretical Computer Science, Vol 55, No. 2, pp. 44–60, Elsevier Science Publishers (2001).

[FSS02] Bernd Finkbeiner, Sriram Sankaranarayanan, and Henny Sipma. Collecting statistics over runtime executions. In Klaus Havelund and Grigore Rosu, editors, *Proc. of Runtime Verification 2002*, volume 70 of *Electronic Notes in Theoretical Computer Science*, pages 36–55. Elsevier Science Publishers, 2002. Accepted for publication in Formal Methods in System Design.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Professional Computing Series. Addison Wesley, 1995.

[HSSS96] Franz Huber, Bernhard Schätz, Alexander Schmidt, and Katharina Spies. AutoFocus - a tool for distributed systems specification. In *Proceedings FTRTFT'96 - Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 1135 in LNCS, pages 467–470. Springer Verlag, 1996.

[Kat93] Shmuel Katz. A superimposition control construct for distributed systems. *ACM Trans. Prog. Lang. Sys.*, 15(2):337–356, April 1993.

[KLM+97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, volume 1241 of *LNCS*. Springer-Verlag, 1997.

[KMP96] Yonit Kesten, Zohar Manna, and Amir Pnueli. Verifying clocked transition systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *LNCS*, pages 13–40. Springer-Verlag, 1996.

[KMP98] Yonit Kesten, Zohar Manna, and Amir Pnueli. Verification of clocked and hybrid systems. In Grzegorz Rozenberg and Frits W. Vaandrager, editors, *Lectures on Embedded Systems*, volume 1494 of *LNCS Tutorial*, pages 4–73. Springer, Heidelberg, 1998.

[MP95] Zohar Manna and Amir Pnueli. Clocked transition systems. In *Proc. of the Intl. Logic and Software Engineering Workshop*, August 1995. Beijing, China.

[Sip03] Henny B. Sipma. A formal model for cross-cutting modular transition systems. In Gary Leavens and Curtis Clifton, editors, *Proceedings of the Workshop on Foundations of Aspect-Oriented Languages (FOAL'03)*, pages 9–16, 2003.

[SSM03] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Petri net analysis using invariant generation. In Nachum Derschowitz, editor, *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 682–701, Taurmina, Italy, 2003. Springer Verlag.

[SSM04] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Non-linear loop invariant generation using Gröbner bases,. In 31^{th} *ACM Symp. Princ. of Prog. Lang.*, pages 318–329, Venice, Italy, January 2004.

[SSRB00] Douglas Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture. Patterns for Concurrent and Networked Objects, Volume 2.* Wiley & Sons, New York, 2000.

[SSS+03] César Sánchez, Sriram Sankaranarayanan, Henny B. Sipma, Ting Zhang, David L. Dill, and Zohar Manna. Event correlation: Language and semantics. In *Embedded Software (EMSOFT)*, volume 2855 of *LNCS*, pages 323–339, 2003.

[SSS+05] César Sánchez, Henny B. Sipma, Venkita Subramonian, Christopher Gill, and Zohar Manna. Thread allocation protocols for distributed and real-time and embedded systems. 2005. Submitted.

[SSSM05] César Sánchez, Henny B. Sipma, Matteo Slanina, and Zohar Manna. Final semantics for event-pattern reactive programs. In *CALCO'05*, 2005. To appear.

[ZSM04a] Ting Zhang, Henny Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints. In *the* 2^{nd} *International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *LNCS*, pages 152–167. Springer-Verlag, 2004.

[ZSM04b] Ting Zhang, Henny Sipma, and Zohar Manna. Term algebras with length function and bounded quantifier alternation. In *the* 17^{th} *International Conference on Theorem Proving in Higher Order Logics (TPHOLs'04)*, volume 3223 of *LNCS*, pages 321–336. Springer-Verlag, 2004.

[ZSM05] Ting Zhang, Henny Sipma, and Zohar Manna. Decision procedures for queues with integer constraints, 2005. Submitted.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
4-29-2005	Final Technical	08-01-2001 to 01-31-2005

4. TITLE AND SUBTITLE

Next Generation Software Development

5a. CONTRACT NUMBER

5b. GRANT NUMBER
DAAD19-01-1- 0723

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Zohar Manna

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Stanford University
Computer Science Department
Stanford, CA 94305-9515

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Mary N. Jackson
Department of the Army
Research Triangle Park
P.O. Box 12211
Research Triangle Pk, NC 27709

10. SPONSOR/MONITOR'S ACRONYM(S)

ONRRO Seattle

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
41923.1-CI

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Under this grant we have studied the development of a scientifically sound basis for software development that builds on widely used pragmatic methods but is firmly grounded in well-established formal domains such as first-order logic and automata theory. To be sufficiently expressive for software systems, the work has focused on methods applicable to infinite-state systems. Traditionally methods for infinite-state systems have been expensive, because they were mainly deductive and thus required guidance by users who were both experts in the application domain and in the verification methodology. Our research has been directed at algorithmic-deductive techniques, including static and runtime analysis techniques that separate the combinatorial reasoning from reasoning about the data. These methods often limit user input to providing abstract system models and application-level guidance, making the interaction more natural to software developers. Constructed proofs hide low-level details; instead, they reason at the most appropriate level of abstraction with respect to the properties to be proved. This characteristic of proofs makes them suitable as system documentation that can evolve with the system. To ensure well-defined semantics, computational models were developed for new computing paradigms, including aspects of publish-subscribe systems and middleware design patterns.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: Unclassified			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE		17	Zohar Manna
					19b. TELEPHONE NUMBER (include area code) (650) 723-4364